正则表达式

- 模块
  - import re
  - re.match() 能够匹配出以xxx开头的字符串
  - 变量名..group()
- 字符串：
  - 保留格式：三引号 """ 内容 """
- 匹配单个字符
  - . 匹配任意1个字符（除了\n）
  - [ ] 匹配[ ]中列举的字符
  - \d 匹配数字，即0-9
  - \D 匹配非数字，即不是数字
  - \s 匹配空白，即 空格，tab键
  - \S 匹配非空白
  - \w 匹配单词字符，即a-z、A-Z、0-9、_
  - \W 匹配非单词字符
- 匹配多个字符
  - * 匹配前一个字符出现0次或者无限次，即可有可无
  - + 匹配前一个字符出现1次或者无限次，即至少有1次
  - ? 匹配前一个字符出现1次或者0次，即要么有1次，要么没有
  - {m} 匹配前一个字符出现m次
  - {m,n} 匹配前一个字符出现从m到n次
- 匹配开头结尾
  - ^ 匹配字符串开头
  - $ 匹配字符串结尾
- 匹配分组
  - | 匹配左右任意一个表达式
  - (ab) 将括号中字符作为一个分组
  - \num 引用分组num匹配到的字符串
  - (?P<name>) 分组起别名 re.match("<(?P<name1>\w*)><(?P<name2>\w*)>.*<(?P=name2)><(?P=name1)>","需要确认是否符合要求的标签")
  - (?P=name) 引用别名为name分组匹配到的字符串
- re模块的高级用法
  - search
    - 查询匹配到字符串，将结果立即返回

```
# 需求：匹配出文章阅读的次数
import re
ret = re.search(r"\d+", "阅读次数为 9999")
ret.group()
```

  - findall
    - 获取所有匹配到的结果

```
# 需求：统计出python、c、c++相应文章阅读的次数
import re
ret = re.findall(r"\d+", "python = 9999, c = 7890, c++ = 12345")
print(ret)
```

  - sub
    - # 将匹配到的阅读次数加1
      - 方法一

```
import re
ret = re.sub(r"\d+", '998', "python = 997")
print(ret)          # python = 998
```

      - 方法二

```
import re
def add(temp):
    strNum = temp.group()
    num = int(strNum) + 1
    return str(num)

ret = re.sub(r"\d+", add, "python = 997")
print(ret)

ret = re.sub(r"\d+", add, "python = 99")
print(ret)
```

    - # 提取文本内容去掉<标签>之类多余的数据 re.sub(r"<[^>]*>| |\n", "", test_str)
  - split 切割字符串 “info:xiaoZhang 33 shandong”

```
import re
ret = re.split(r ":| ", "info:xiaoZhang 33 shandong")
print(ret)      # ['info', 'xiaoZhang', '33', 'shandong']
```

- python贪婪和非贪婪
  - 贪婪 Python里数量词默认是贪婪的（在少数语言里也可能是默认非贪婪），总是尝试匹配尽可能多的字符
  - 非贪婪
    - 非贪婪则相反，总是尝试匹配尽可能少的字符
    - **在"*","?","+","{m,n}"后面加上？，使贪婪变成非贪婪**
- r的作用
  - 代表一个原生字符串 Python里的原生字符串很好地解决了这个问题，有了原生字符串，你再也不用担心是不是漏写了反斜杠，写出来的表达式也更直观
  - 备注：正则表达式中，"\" 作为转义符 其中\a响铃 \b delete 有专属意义，想保持没有歧义加一个 “r”

Python中字符串前面加上 r 表示原生字符串，

与大多数编程语言相同，正则表达式里使用"\"作为转义字符，这就可能造成反斜杠困扰。假如你需要匹配文本中的字符"\"，那么使用编程语言表示的正则表达式里将需要4个反斜杠"\\"：前两个和后两个分别用于在编程语言里转义成反斜杠，转换成两个反斜杠后再在正则表达式里转义成一个反斜杠。

- 转义符：反斜杠
  - \ 键盘中唯一的一个